Machine Learning & Data Mining

Lecture 6

Corina Besliu

Technical University of Moldova

November 17, 2021

Outline

- Logistic Regression (for Classification) Model
- Decision Tree and Random Forest Classification Models
- K-Nearest Neighbors (K-NN) Model

Outline

- Logistic Regression (for Classification) Model
- Decision Tree and Random Forest Classification Models
- K-Nearest Neighbors (K-NN) Model

Outline

- Logistic Regression (for Classification) Model
- Decision Tree and Random Forest Classification Models
- K-Nearest Neighbors (K-NN) Model

Logistic Regression

Logistic Regression

Linear Regression:

- **Simple:**

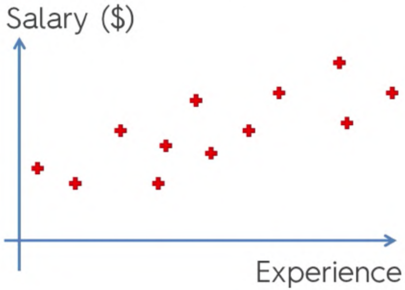
  $y = b_0 + b_1 * x$

- **Multiple:**

  $y = b_0 + b_1 * x_1 + \ldots + b_n * x_n$

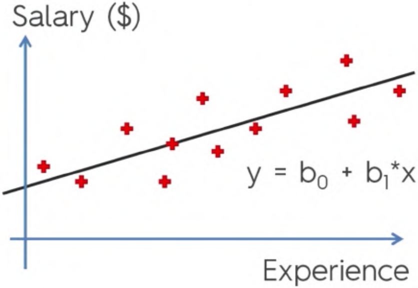
Logistic Regression

We know this:

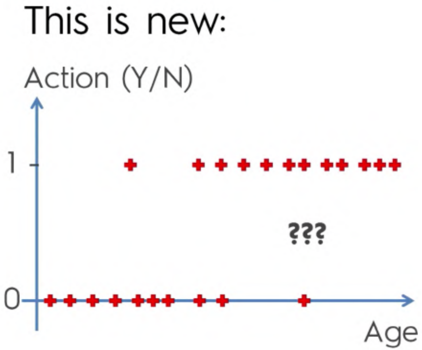


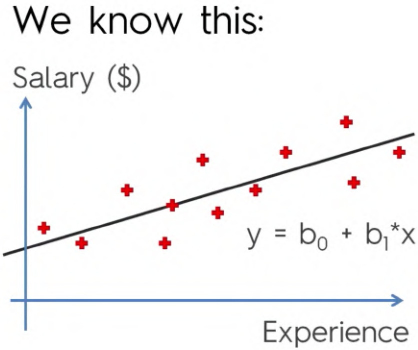
Logistic Regression

We know this:



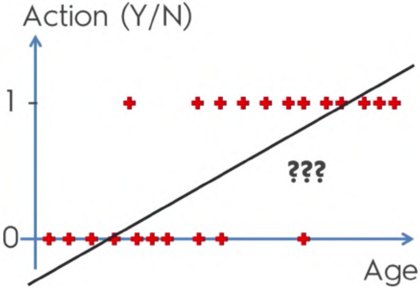
Logistic Regression



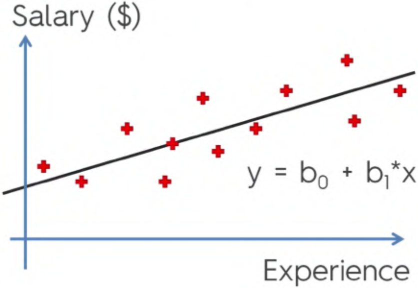


Logistic Regression

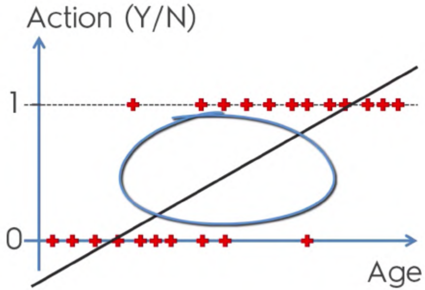
This is new:



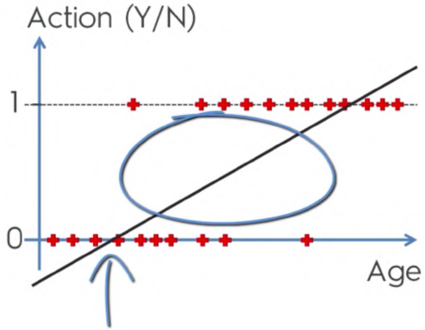
We know this:



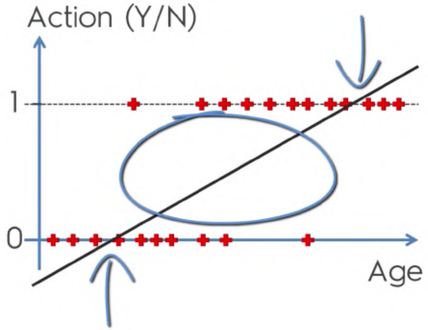
Logistic Regression



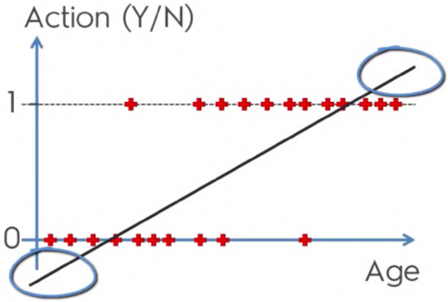
Logistic Regression



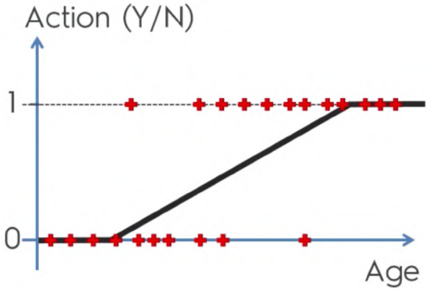
Logistic Regression



Logistic Regression

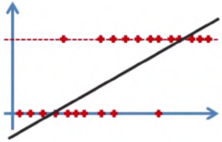


Logistic Regression



Logistic Regression

$$y = b_0 + b_1 * x$$

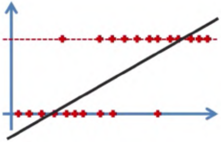


Logistic Regression

$$y = b_0 + b_1 * x$$

Sigmoid Function

$$p = \frac{1}{1 + e^{-y}}$$



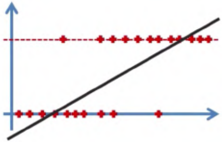
Logistic Regression

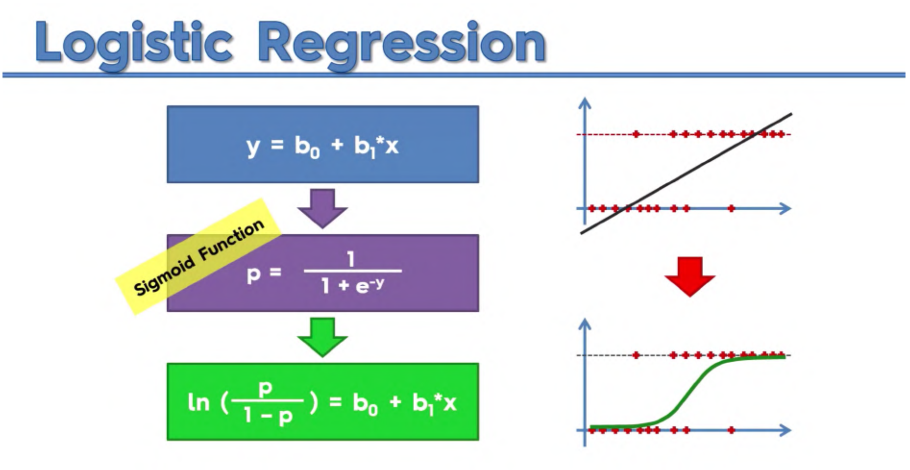
$$y = b_0 + b_1 * x$$

Sigmoid Function

$$p = \frac{1}{1 + e^{-y}}$$

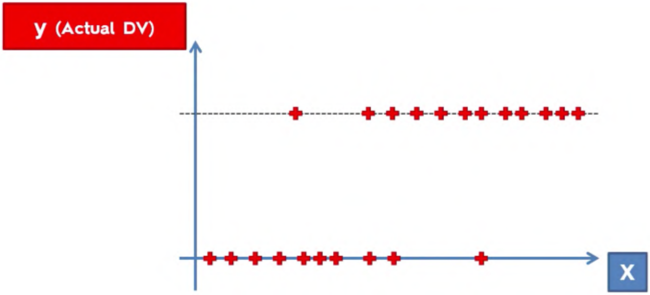
$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 * x$$



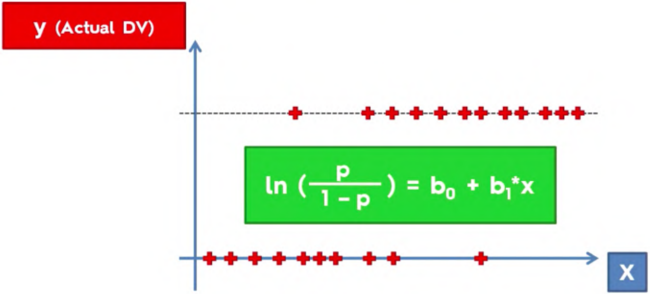

Logistic Regression
$y = b_0 + b_1 {}^* x$
Sigmoid Function
$p = \frac{1}{1 + e^{-y}}$
$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 {}^* x$


Logistic Regression
WHAT JUST
HAPPENED
???

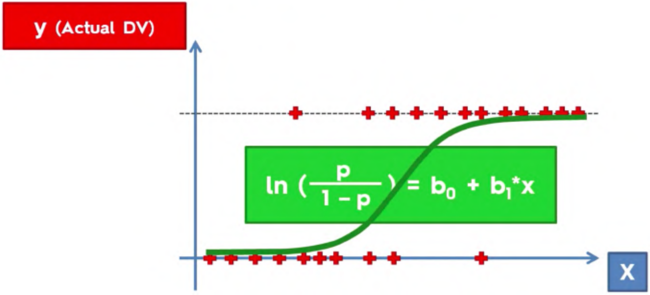
Logistic Regression



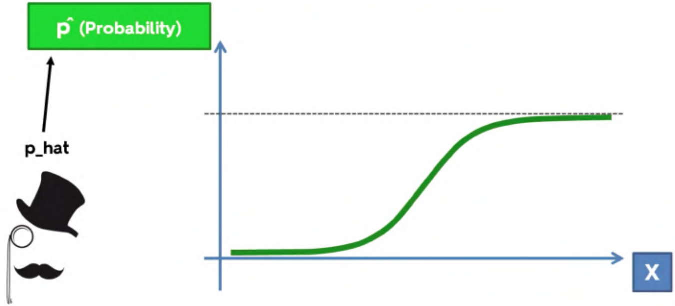
Logistic Regression



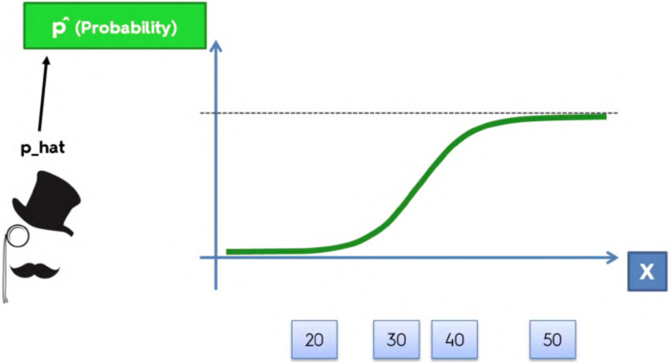
Logistic Regression



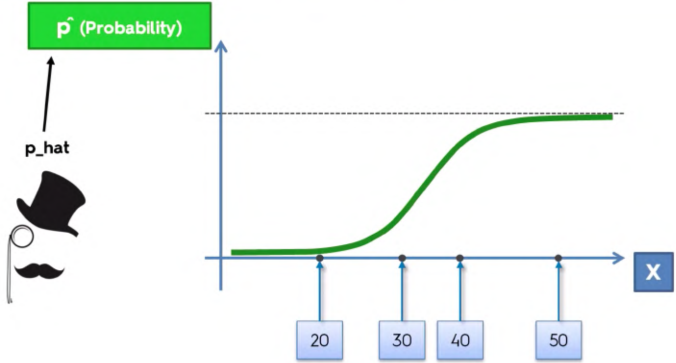
Logistic Regression



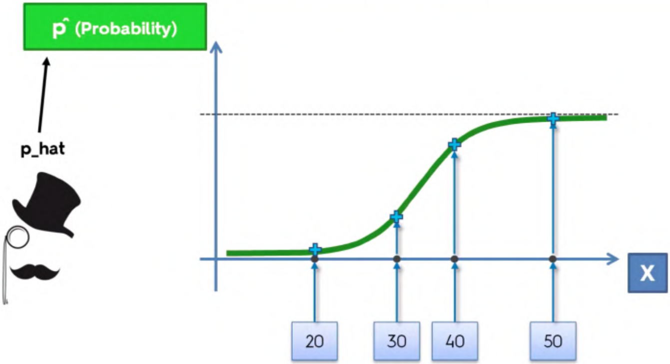
Logistic Regression



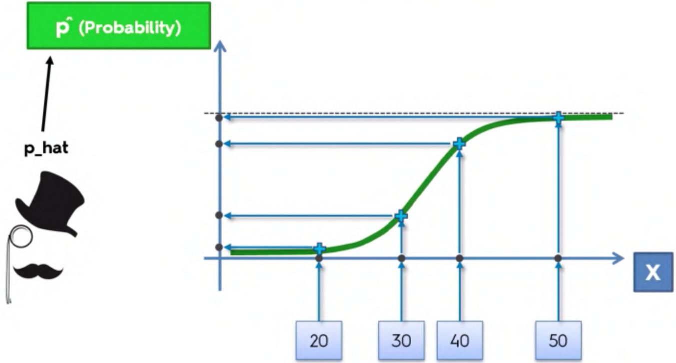
Logistic Regression



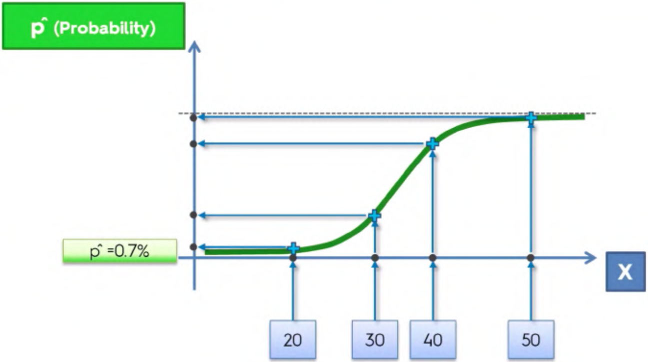
Logistic Regression



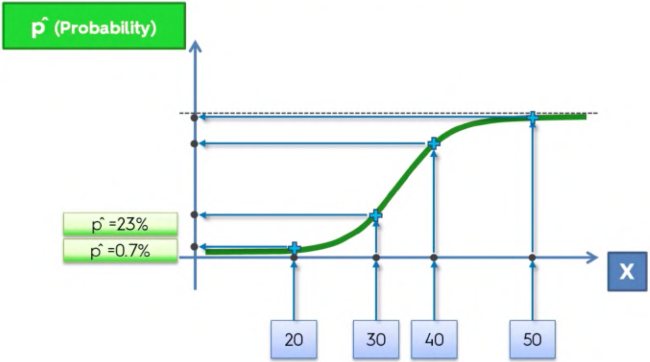
Logistic Regression



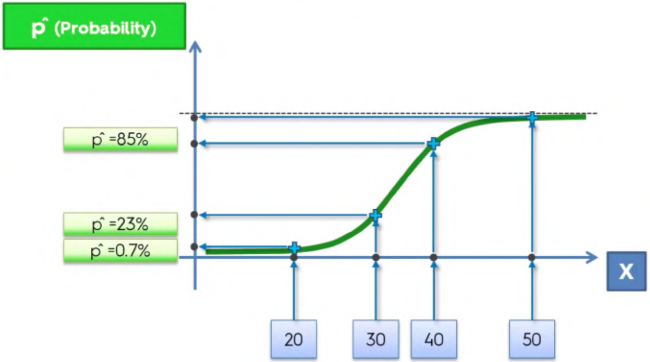
Logistic Regression



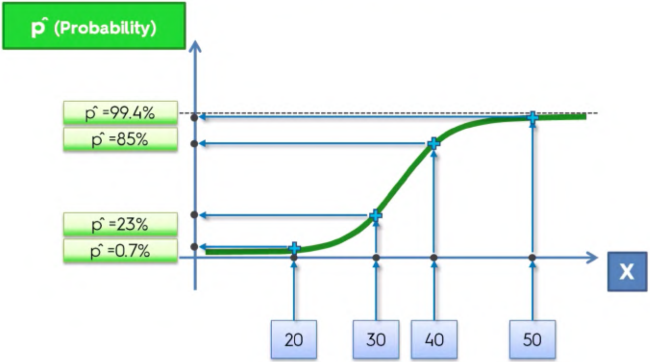
Logistic Regression

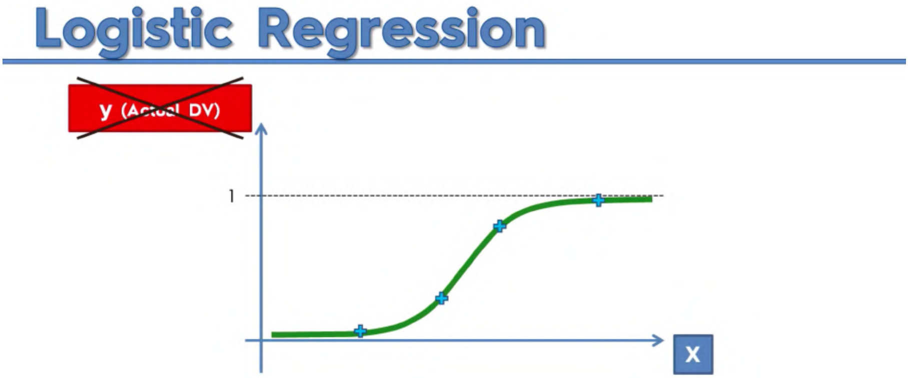


Logistic Regression

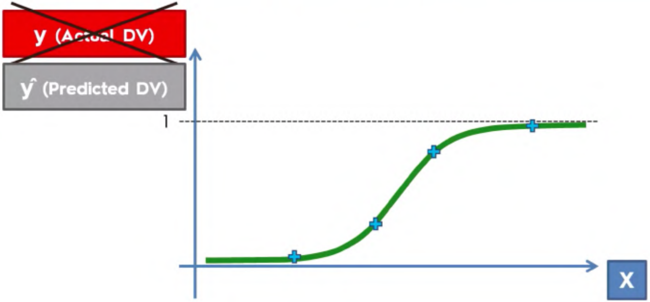


Logistic Regression

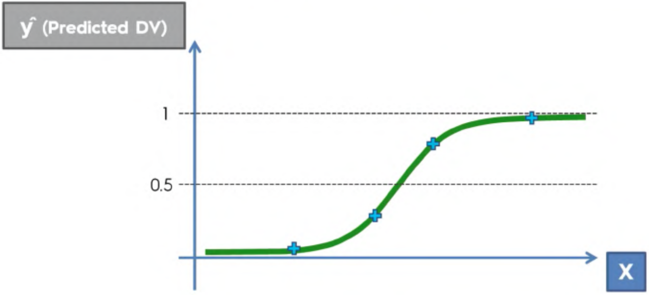



Logistic Regression
y (Actual DV)
1
X

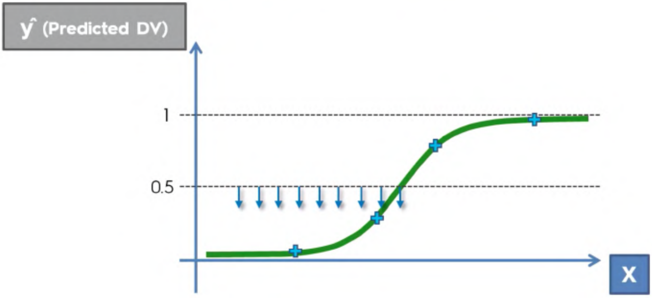
Logistic Regression



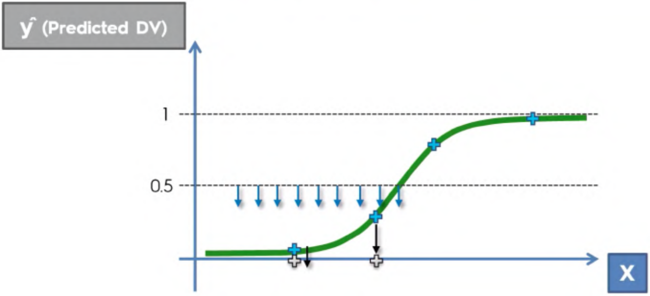
Logistic Regression



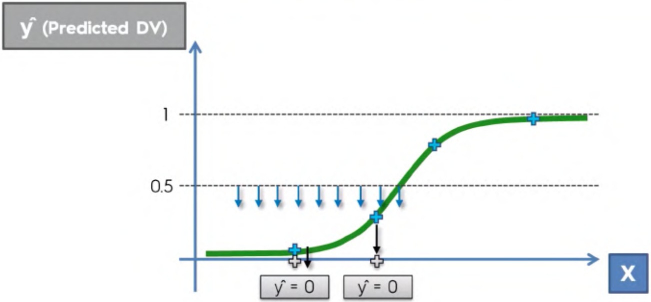
Logistic Regression



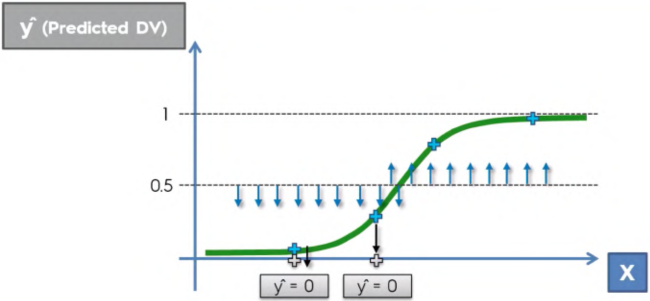
Logistic Regression



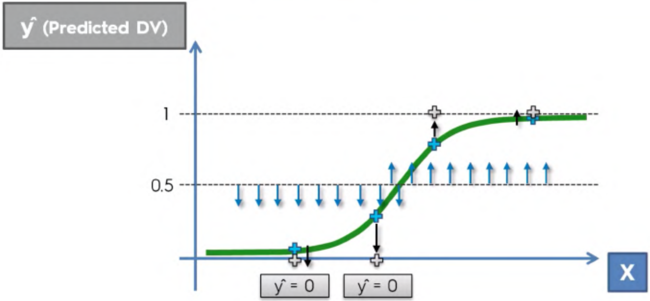
Logistic Regression



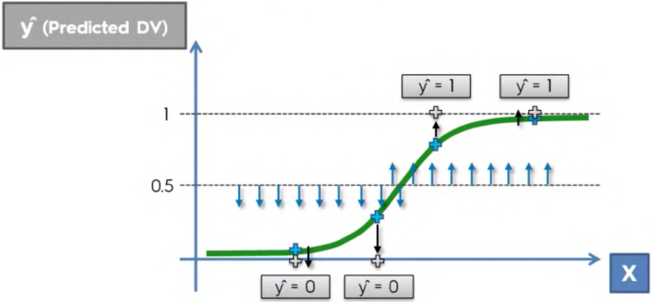
Logistic Regression



Logistic Regression



Logistic Regression



Pros & Cons of Logistic Regression

PROS

- Easy to implement, interpret, and very efficient to train.
- Can predict probabilities
- Makes no assumptions about distributions of classes in feature space.
- Can easily extend to multiple classes (multinomial regression).
- Fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- Less inclined to over-fitting but it can overfit in high dimensional datasets.

One may consider Regularization (e.g. Ridge and Laso Regression: L1 and L2.
see here about L1 & L2: `https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-lea`

Pros & Cons of Logistic Regression

PROS

- Easy to implement, interpret, and very efficient to train.
- Can predict probabilities
- Makes no assumptions about distributions of classes in feature space.
- Can easily extend to multiple classes (multinomial regression).
- Fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- Less inclined to over-fitting but it can overfit in high dimensional datasets.

One may consider Regularization (e.g. Ridge and Laso Regression: L1 and L2. *see here about L1 & L2:* `https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-lea`

Pros & Cons of Logistic Regression

PROS

- Easy to implement, interpret, and very efficient to train.
- Can predict probabilities
- Makes no assumptions about distributions of classes in feature space.
- Can easily extend to multiple classes (multinomial regression).
- Fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- Less inclined to over-fitting but it can overfit in high dimensional datasets.

One may consider Regularization (e.g. Ridge and Laso Regression: L1 and L2. *see here about L1 & L2:* `https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-lea`

Pros & Cons of Logistic Regression

PROS

- Easy to implement, interpret, and very efficient to train.
- Can predict probabilities
- Makes no assumptions about distributions of classes in feature space.
- Can easily extend to multiple classes (multinomial regression).
- Fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- Less inclined to over-fitting but it can overfit in high dimensional datasets.

One may consider Regularization (e.g. Ridge and Laso Regression: L1 and L2.
see here about L1 & L2: `https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-lea`

Pros & Cons of Logistic Regression

PROS

- Easy to implement, interpret, and very efficient to train.
- Can predict probabilities
- Makes no assumptions about distributions of classes in feature space.
- Can easily extend to multiple classes (multinomial regression).
- Fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- Less inclined to over-fitting but it can overfit in high dimensional datasets.

One may consider Regularization (e.g. Ridge and Laso Regression: L1 and L2.

see here about L1 & L2: `https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-lea`

Pros & Cons of Logistic Regression

PROS

- Easy to implement, interpret, and very efficient to train.
- Can predict probabilities
- Makes no assumptions about distributions of classes in feature space.
- Can easily extend to multiple classes (multinomial regression).
- Fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- Less inclined to over-fitting but it can overfit in high dimensional datasets.

One may consider Regularization (e.g. Ridge and Laso Regression: L1 and L2. *see here about L1 & L2:* `https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-lea`

Pros & Cons of Logistic Regression

PROS

- Easy to implement, interpret, and very efficient to train.
- Can predict probabilities
- Makes no assumptions about distributions of classes in feature space.
- Can easily extend to multiple classes (multinomial regression).
- Fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- Less inclined to over-fitting but it can overfit in high dimensional datasets.

One may consider Regularization (e.g. Ridge and Laso Regression: L1 and L2. *see here about L1 & L2:* `https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-lea`

Pros & Cons of Logistic Regression

PROS

- Easy to implement, interpret, and very efficient to train.
- Can predict probabilities
- Makes no assumptions about distributions of classes in feature space.
- Can easily extend to multiple classes (multinomial regression).
- Fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable.
- Less inclined to over-fitting but it can overfit in high dimensional datasets.

One may consider Regularization (e.g. Ridge and Laso Regression: L1 and L2. *see here about L1 & L2:* `https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-lea`

Pros & Cons of Logistic Regression

CONS

- Implementation with more than 2 classes is cumbersome.
- The assumption of linearity between the dependent variable and the independent variables.
  - Non-linear problems cant be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
- requires average or no multicollinearity between independent variables.
- It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.

Pros & Cons of Logistic Regression

CONS

- Implementation with more than 2 classes is cumbersome.
- The assumption of linearity between the dependent variable and the independent variables.
  - Non-linear problems cant be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
- requires average or no multicollinearity between independent variables.
- It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.

Pros & Cons of Logistic Regression

CONS

- Implementation with more than 2 classes is cumbersome.
- The assumption of linearity between the dependent variable and the independent variables.
  - Non-linear problems cant be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
- requires average or no multicollinearity between independent variables.
- It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.

Pros & Cons of Logistic Regression

CONS

- Implementation with more than 2 classes is cumbersome.
- The assumption of linearity between the dependent variable and the independent variables.
  - Non-linear problems cant be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
- requires average or no multicollinearity between independent variables.
- It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.

Pros & Cons of Logistic Regression

CONS

- Implementation with more than 2 classes is cumbersome.
- The assumption of linearity between the dependent variable and the independent variables.
  - Non-linear problems cant be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
- requires average or no multicollinearity between independent variables.
- It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.

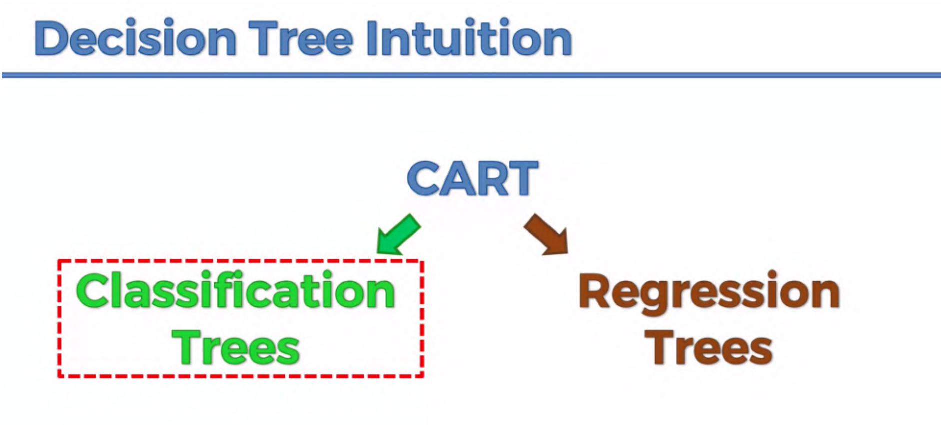
Decision Tree Classification Model

Decision Tree Intuition

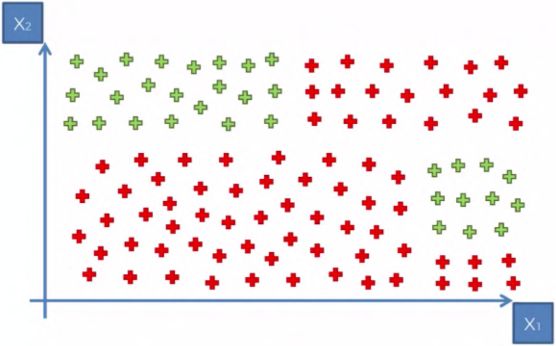
CART

Classification Trees

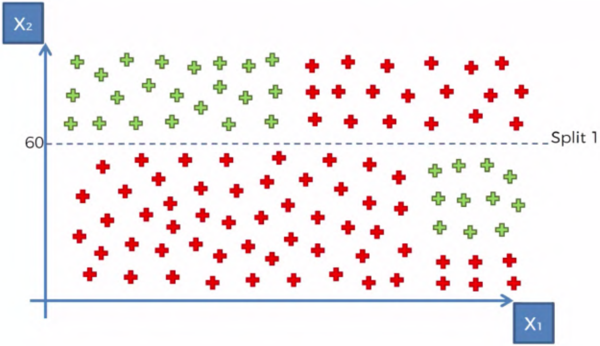
Regression Trees


Decision Tree Intuition
CART
Classification Trees
Regression Trees

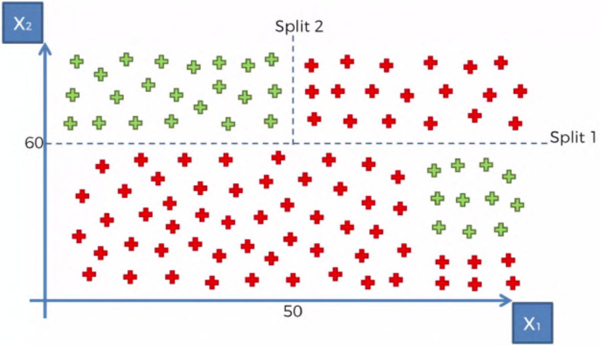
Decision Tree Intuition



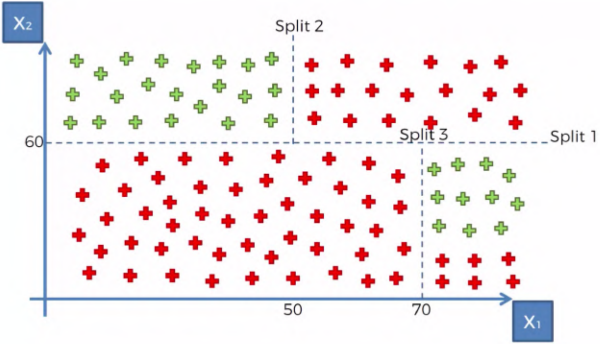
Decision Tree Intuition



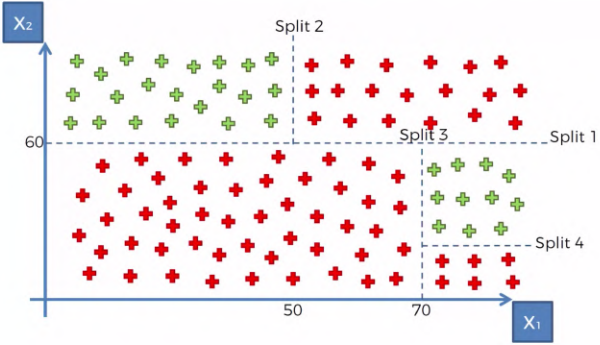
Decision Tree Intuition



Decision Tree Intuition



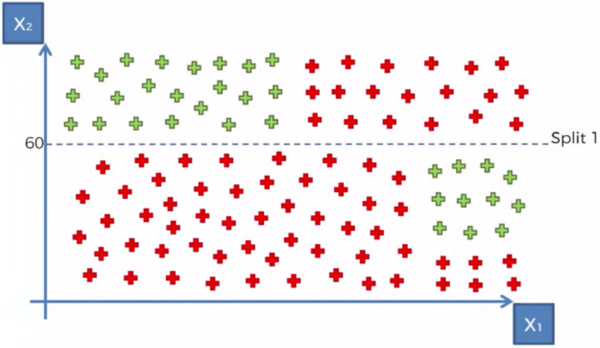
Decision Tree Intuition



Decision Tree Intuition

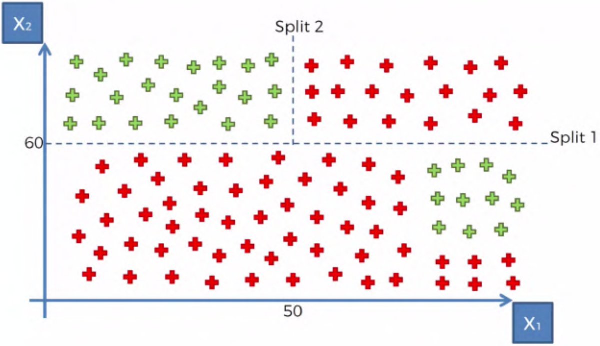
Rewind...

Decision Tree Intuition

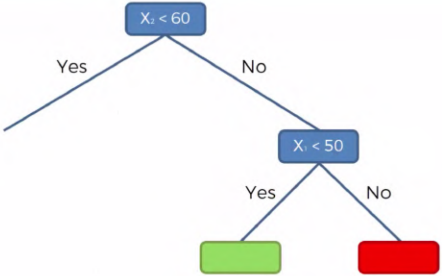


Split 1

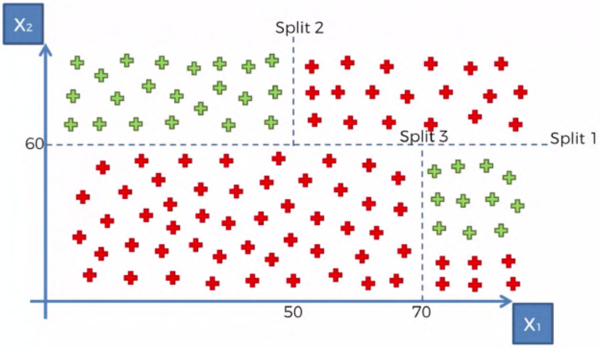
Decision Tree Intuition



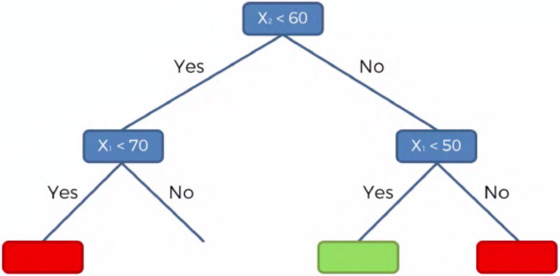
Split 2



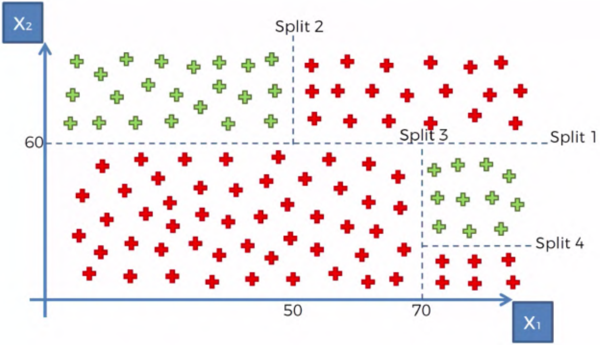
Decision Tree Intuition



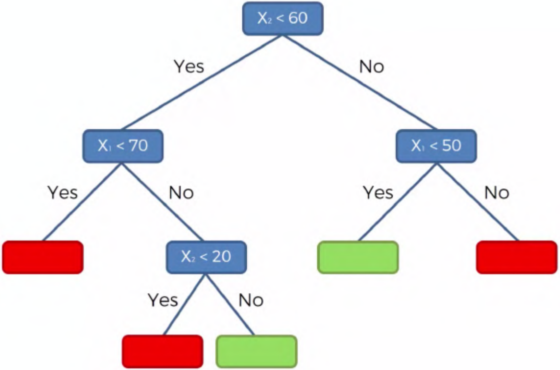
Split 3



Decision Tree Intuition



Split 4



Making the Splits

How are the splits done?

Decision tree algorithms use information gain to split a node.

Mainly two criteria: Gini index and entropy are used for calculating information gain.

Both gini and entropy are measures of impurity of a node. A node having multiple classes is impure whereas a node having only one class is pure.

Entropy in statistics is analogous to entropy in thermodynamics where it signifies disorder. If there are multiple classes in a node, there is disorder in that node.

Here are some videos with examples of how these measures are computed:
Gini Index: `https://www.youtube.com/watch?v=_L39rN6gz7Y`
Entropy: `https://www.youtube.com/watch?v=1IQOtJ4NI_0`
Information Gain: `https://www.youtube.com/watch?v=FuTRucXB9rA`

Making the Splits

How are the splits done?

Decision tree algorithms use information gain to split a node.

Mainly two criteria: Gini index and entropy are used for calculating information gain.

Both gini and entropy are measures of impurity of a node. A node having multiple classes is impure whereas a node having only one class is pure.

Entropy in statistics is analogous to entropy in thermodynamics where it signifies disorder. If there are multiple classes in a node, there is disorder in that node.

Here are some videos with examples of how these measures are computed:
Gini Index: `https://www.youtube.com/watch?v=_L39rN6gz7Y`
Entropy: `https://www.youtube.com/watch?v=1IQOtJ4NI_0`
Information Gain: `https://www.youtube.com/watch?v=FuTRucXB9rA`

Making the Splits

How are the splits done?

Decision tree algorithms use information gain to split a node.

Mainly two criteria: Gini index and entropy are used for calculating information gain.

Both gini and entropy are measures of impurity of a node. A node having multiple classes is impure whereas a node having only one class is pure.

Entropy in statistics is analogous to entropy in thermodynamics where it signifies disorder. If there are multiple classes in a node, there is disorder in that node.

Here are some videos with examples of how these measures are computed:
Gini Index: `https://www.youtube.com/watch?v=_L39rN6gz7Y`
Entropy: `https://www.youtube.com/watch?v=1IQOtJ4NI_0`
Information Gain: `https://www.youtube.com/watch?v=FuTRucXB9rA`

Making the Splits

How are the splits done?

Decision tree algorithms use information gain to split a node.

Mainly two criteria: Gini index and entropy are used for calculating information gain.

Both gini and entropy are measures of impurity of a node. A node having multiple classes is impure whereas a node having only one class is pure.

Entropy in statistics is analogous to entropy in thermodynamics where it signifies disorder. If there are multiple classes in a node, there is disorder in that node.

Here are some videos with examples of how these measures are computed:
Gini Index: https://www.youtube.com/watch?v=_L39rN6gz7Y
Entropy: https://www.youtube.com/watch?v=1IQOtJ4NI_0
Information Gain: https://www.youtube.com/watch?v=FuTRucXB9rA

Making the Splits

How are the splits done?

Decision tree algorithms use information gain to split a node.

Mainly two criteria: Gini index and entropy are used for calculating information gain.

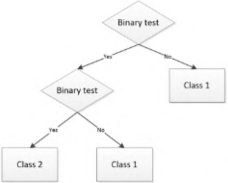
Both gini and entropy are measures of impurity of a node. A node having multiple classes is impure whereas a node having only one class is pure.

Entropy in statistics is analogous to entropy in thermodynamics where it signifies disorder. If there are multiple classes in a node, there is disorder in that node.

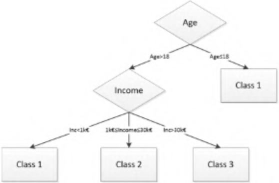
Here are some videos with examples of how these measures are computed:
Gini Index: `https://www.youtube.com/watch?v=_L39rN6gz7Y`
Entropy: `https://www.youtube.com/watch?v=1IQOtJ4NI_0`
Information Gain: `https://www.youtube.com/watch?v=FuTRucXB9rA`

Decisions trees can be categorized according to three criteria:

- The type of data: Numerical, Categorical, Mixed
- The type of nodes: Binary leaves, multiple leaves
- The overall shape of the tree

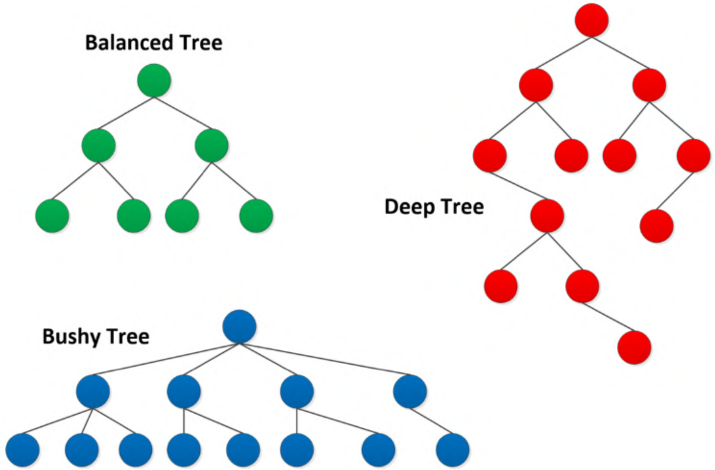


(c) Example of a binary tree



(d) Example of a numerical non-binary tree

Current Slide is a Courtesy of Dr. N. Grozavu



Current Slide is a Courtesy of Dr. N. Grozavu

- Deep trees are usually very biased, can't generalize much outside of their training set and are difficult to interpret.

Setting the right Depth for your tree

Most Decision trees algorithms will allow you to choose the maximum depth of your tree.

- How Deep is too deep will depend on the complexity of the problem
- Deeper trees tend towards overfitting, while less deep trees will tend towards underfitting.
- The best option is to start from a deep tree and to prune it in a way that minimizes the error on the training set.

- While balanced-trees are usually the most preferable option, bushy trees should not be frowned upon in problems with a lot of classes, or when they can help reducing the depth of the tree.

Slide is a Courtesy of Dr. N. Grozavu

Pros & Cons of Decision Trees

Pros

- Intuitive, easy to understand and to use
- Build comprehensive models
- The most commonly classifier for decision making
- Can learn in a single sweep

Cons

- The process to build the tree is complex
- There are always several possible trees
- Choosing the depth of the tree is a complex decision
- Does not work well with datasets that have too many attributes.

Current Slide is a Courtesy of Dr. N. Grozavu

Decision Trees

- **Old Method**

Decision Trees

- Old Method
- Reborn with upgrades

Decision Trees

- Old Method
- Reborn with upgrades
- Random Forest
- Gradient Boosting
- etc.

Random Forest Classification Model

Random Forest Intuition

STEP 1: Pick at random K data points from the Training set.

⇩

STEP 2: Build the Decision Tree associated to these K data points.

⇩

STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2

⇩

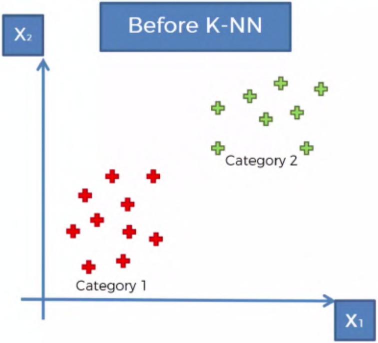
STEP 4: For a new data point, make each one of your Ntree trees predict the category to which the data points belongs, and assign the new data point to the category that wins the majority vote.

Random Forest Intuition

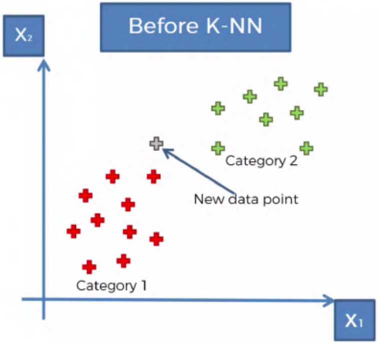
//www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf

K-NN Model

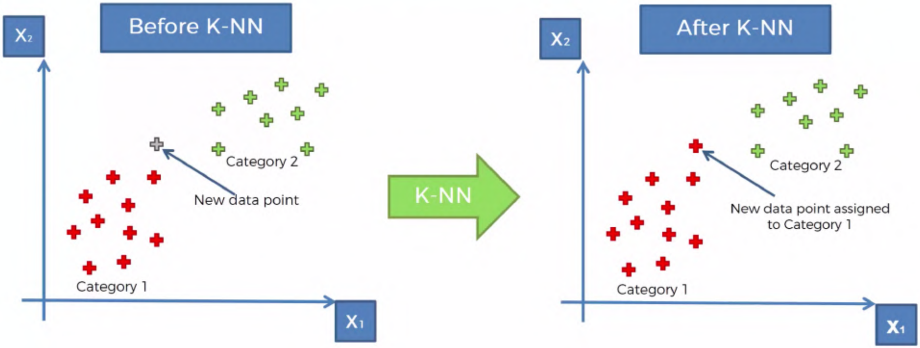
What K-NN does for you



What K-NN does for you



What K-NN does for you



How did it do that ?

STEP 1: Choose the number K of neighbors

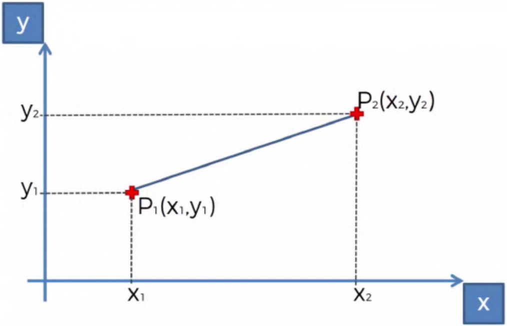
How did it do that ?

STEP 1: Choose the number K of neighbors

⇩

STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance

Euclidean Distance



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

How did it do that ?

STEP 1: Choose the number K of neighbors

⇩

STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance

⇩

STEP 3: Among these K neighbors, count the number of data points in each category

How did it do that ?

STEP 1: Choose the number K of neighbors

⇩

STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance

⇩

STEP 3: Among these K neighbors, count the number of data points in each category

⇩

STEP 4: Assign the new data point to the category where you counted the most neighbors

⇩

Your Model is Ready

K-NN algorithm

STEP 2: Take the K = 5 nearest neighbors of the new data point, according to the Euclidean distance

K-NN algorithm

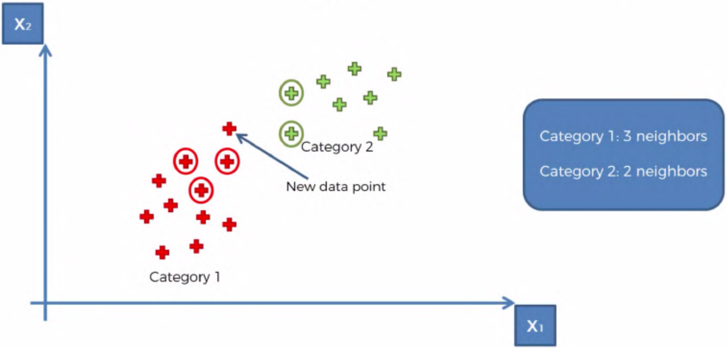
STEP 3: Among these K neighbors, count the number of data points in each category

Category 1: 3 neighbors

Category 2: 2 neighbors

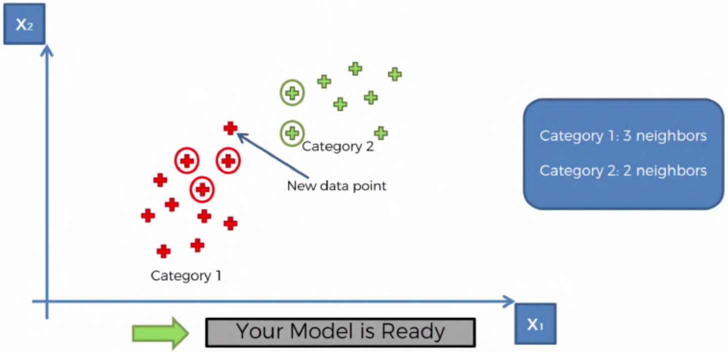
K-NN algorithm

STEP 4: Assign the new data point to the category where you counted the most neighbors



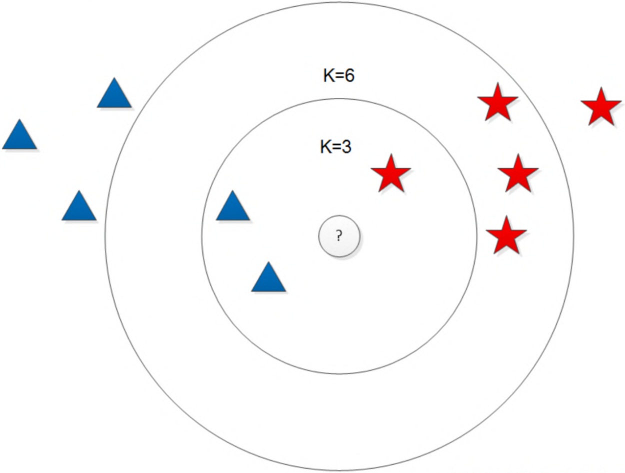
K-NN algorithm

STEP 4: Assign the new data point to the category where you counted the most neighbors



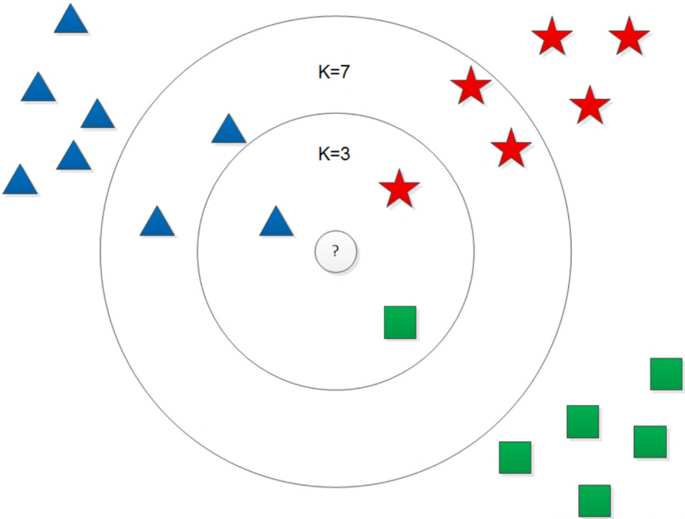
K is a critical parameter that can render the algorithm quickly unstable:

- Depending on the K, the class changes completely.



Slide is a Courtesy of Dr. N. Grozavu

With more than 2 classes, things can quickly become complicated ...



Slide is a Courtesy of Dr. N. Grozavu

- Because the distance between instances is based on all the attributes, less relevant attributes and even the irrelevant ones are used in the classification of a new instance.
- Because the algorithm delays all processing until a new classification/prediction is required, significant processing is needed to make the prediction.

The **Weighted Nearest Neighbors** solves 2 of the previous problems by adding a weight w_k to each neighbor.

Examples:

$$w_k = \begin{cases} \frac{1}{k} & \text{if } k \leq K \\ 0 & \text{if } k > K \end{cases}$$

$$w_k = \begin{cases} \frac{1}{dist} & \text{if } k \leq K \\ 0 & \text{if } k > K \end{cases}$$

Remark

The real Weighted Nearest Neighbors classifier uses a much more complex weight system that satisfies $\sum_{n=1}^{N} w_{ni} = 1$.

Pros & Cons of the K-NN Model

Pros

- Very simple and intuitive
- Low Complexity
- Great results with well-behaved classes

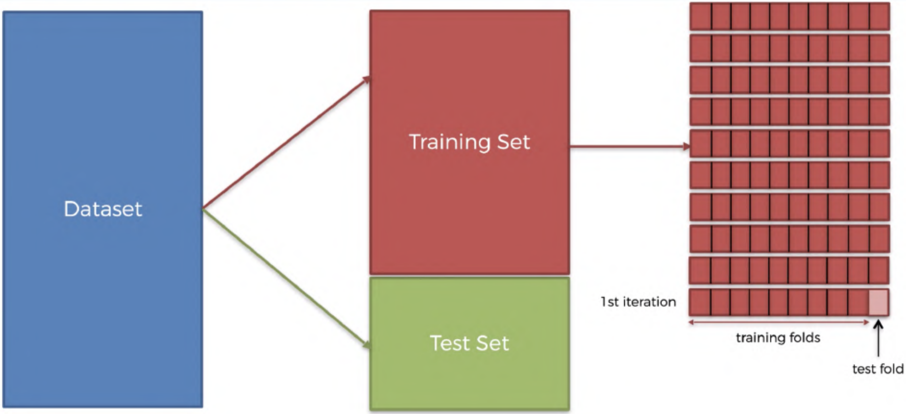
Cons

- No model: No way to properly describe each class. No possibility to re-use the knowledge
- Does not scale well because it requires to store all the training set
- Critical choice of the parameter K
- Ill-adapted for categorical data

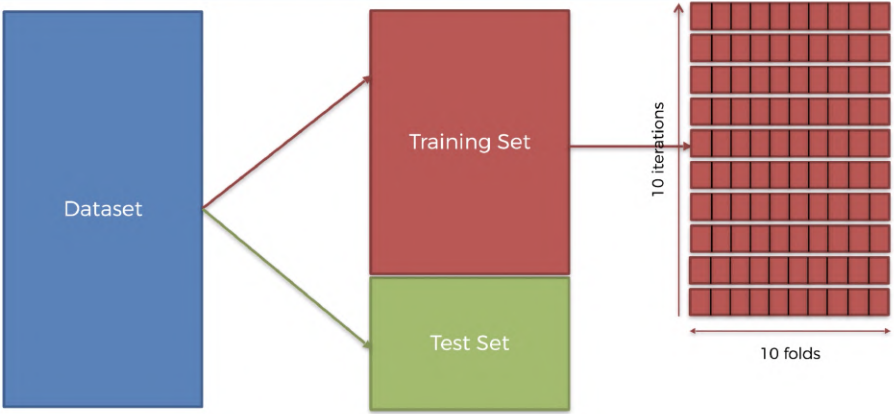
Current Slide is a Courtesy of Dr. N. Grozavu

K-Fold Cross Validation

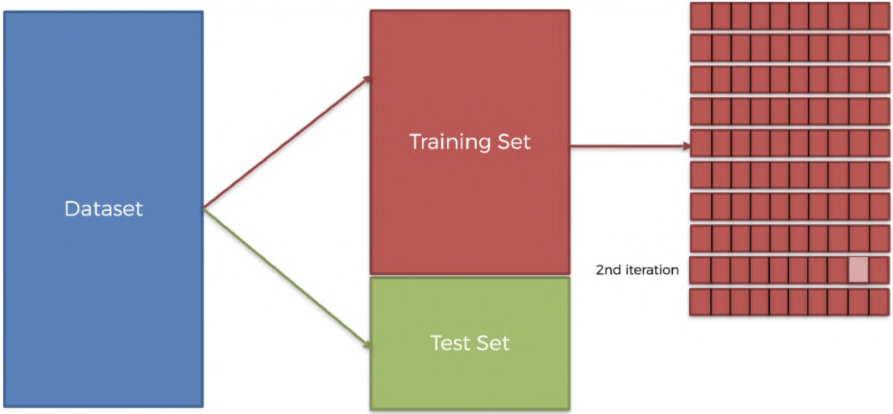
k-Fold Cross Validation

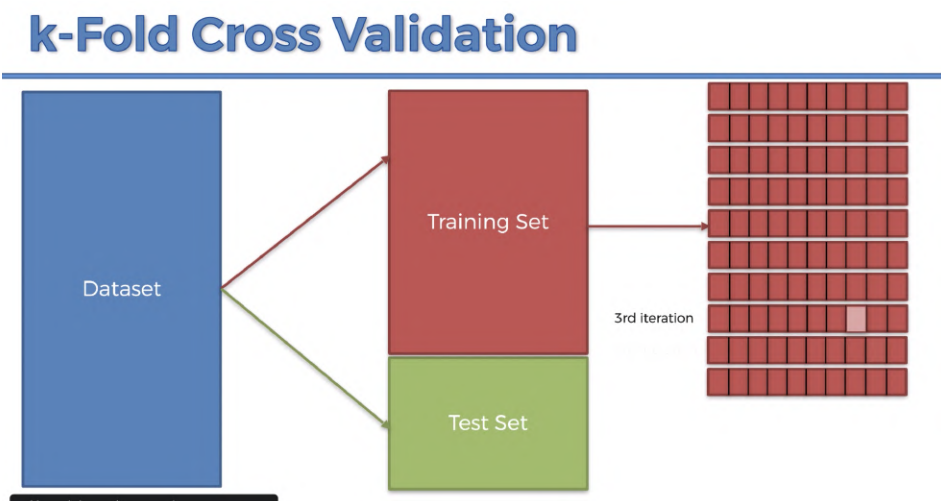


k-Fold Cross Validation

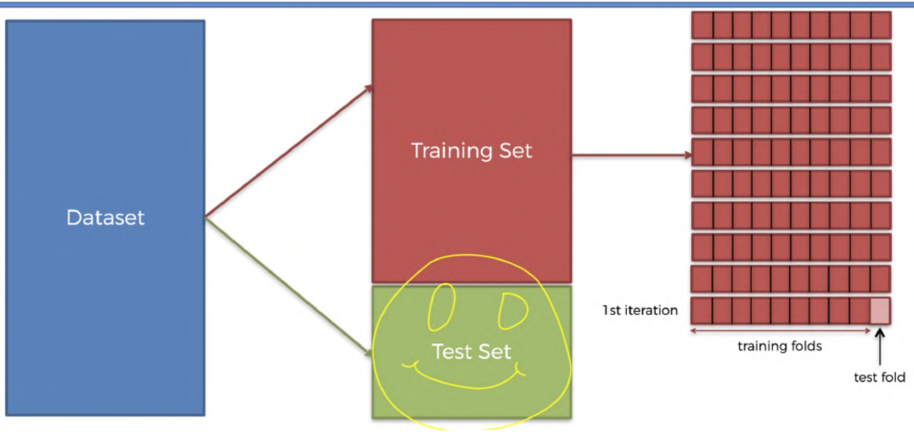


k-Fold Cross Validation




k-Fold Cross Validation
Dataset
Training Set
Test Set
3rd iteration

k-Fold Cross Validation



Let's get Started!

Access Google Colaboratory through your Gmail account